Project Assignment
System Identification
2022-2023

Professor:
Busoniu Lucian

Students:
Oprea Sergiu-Daniel
Radu Victor
Ștefănuți Mihai

Group: 30332
Index of the datafile: 1

CONTENTS

Part 1. Time series modeling using Fourier basis functions

- **Introduction**

  The aim of this project is to model and approximate the trend and interest in the products offered by a building engineering parts store. We will use the linear regression method in order to reach this objective. Our input dataset consists of the number of product units sold by the store as a function of time. We can use this information to firstly create a reliable approximator for our real-world model, followed by applying the linear regression algorithm in order to obtain the values corresponding to trend and interest. For the final step, we will compare the results with the value of the mean square error we obtained earlier in the process, so we can find the best approximation for our initial model.

- **Introducing the approximator structure:**

  In order to approximate our model, we're using an approximator structure composed of a Fourier function as a basis function, and an unknown parameter vector. The formula for the Fourrier basis function is written below:

$$\sum_{i=1}^{m} \left[a_i \cos\left(\frac{2\pi i\, k}{P}\right) + b_i \sin\left(\frac{2\pi i\, k}{P}\right) \right]$$

  This function has a configurable number of terms, which is done by changing the value of the variable m, and we're taking the value of the variable P to be equal to 12, as we're studying the input data over a time period of 12 months. Each Fourier term will give two basis functions, one with cos and one with sin, corresponding to different frequencies. By adding the linear trend component to this basis function, we obtain the values of the regressor phi.

After successfully determining the phi vector, we can compute the unknown parameter vector, by solving a linear system of equations written in a matrix form, as demonstrated below:

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} \varphi_1(1) & \varphi_2(1) & \dots & \varphi_n(1) \\ \varphi_1(2) & \varphi_2(2) & \dots & \varphi_n(2) \\ \dots & \dots & \dots & \dots \\ \varphi_1(N) & \varphi_2(N) & \dots & \varphi_n(N) \end{bmatrix} \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \dots \\ \theta_n \end{bmatrix}$$

In the end, we obtain the approximated output vector by multiplying the two matrices, denoted with the letters phi and theta.
For example, this is what the approximator looks like for m = 1:

$$\begin{aligned} \hat{y}(k) &= t_0 + t_1 k + a_1 \cos\left(\frac{2\pi\, k}{12}\right) + b_1 \sin\left(\frac{2\pi\, k}{12}\right) \\ &= \varphi^\top(k)\theta = \left[1,\ k,\ \cos\left(\frac{2\pi k}{12}\right),\ \sin\left(\frac{2\pi k}{12}\right)\right] \cdot \begin{bmatrix} t_0 \\ t_1 \\ a_1 \\ b_1 \end{bmatrix} \end{aligned}$$

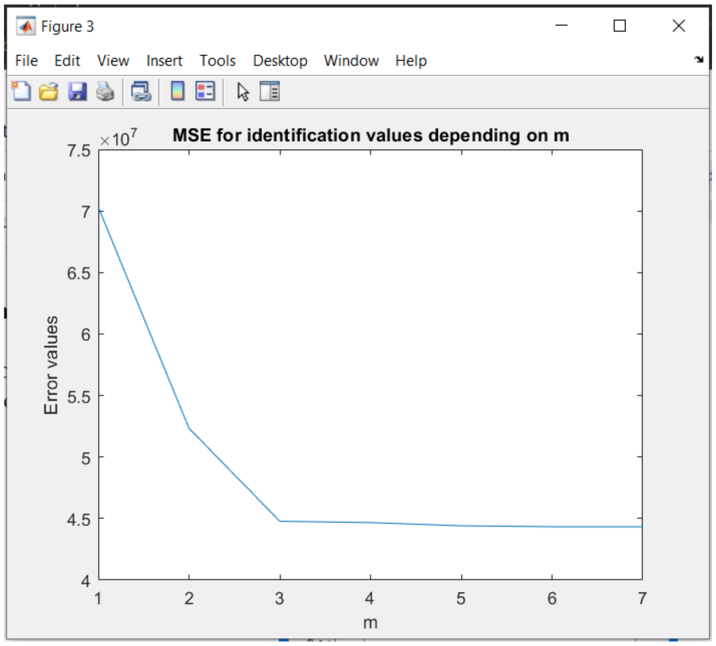
The next thing that should be done is to check how close our approximations are to the real output data, and tune the variables of the approximator structure accordingly.

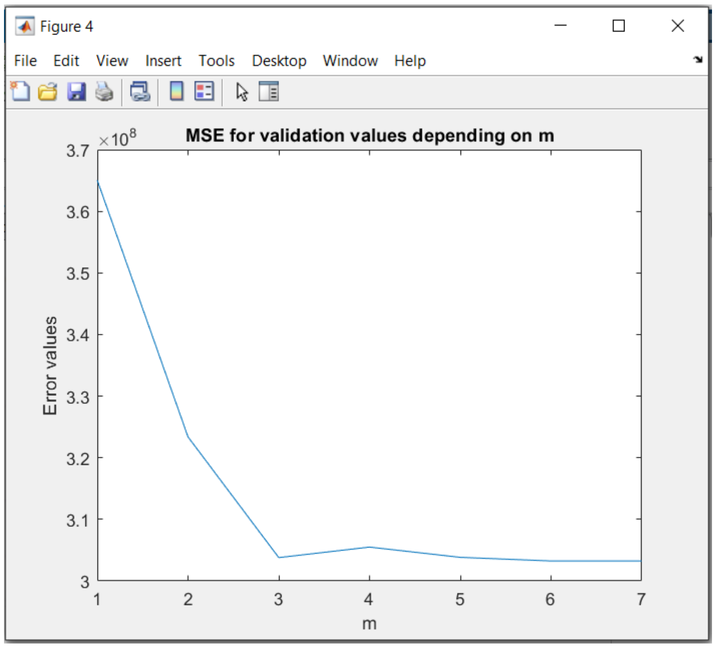
- **The algorithm**

The algorithm loads a dataset that contains an input vector and a corresponding output vector. The first 80% of this dataset is used for identification, while the other 20% is used for validation. Because the data is monthly and at an at most yearly periodicity, the default period will equal 12. Because the number of Fourier terms “m” is configurable from 1 to 7, the MSEs (as functions of “m”) at the identification and validation stages will be stored in matrices having 1 row and 7 columns, initialized as having all elements equal to 0. Starting from m equal to 1, the ϕ transposed matrix is initialized as having the number of rows equal to the length of the time of the identification set and the number of columns equal to 2 + 2 * m, also initialized as having all elements equal to 0. Then, in a loop having the number of iterations equal to the length of the identification time, the first column of the ϕ transposed matrix will be given the value "1", the second column of the matrix will be given the value of the identification time at the index equal to the row, and there will be another loop in which the rest of the ϕ transposed matrix will be populated, according to the formula. From that, Θ, the approximated output and the MSE are computed, the last one being stored in the MSE identification vector. Then the process of populating the ϕ transposed matrix and finding Θ, the approximated output and the MSE is repeated for the validation data, and the MSE is stored in the MSE validation vector. This entire process is repeated for m = 2, …, 7 using an iterative loop, in order to fully populate the MSE identification and validation vectors. Finally, the approximated outputs at the identification and validation stages are plotted alongside their respective outputs from the dataset, and the MSE vectors are each individually plotted as functions of m, in order to show which value of m will result in the lowest MSE and for which value of m is the difference between the MSEs at identification and validation the smallest (It is worth acknowledging that the algorithm only plots the approximated outputs at the final iteration of the "m" loop, where m = 7 and the Fourier Series is at its most extensive).

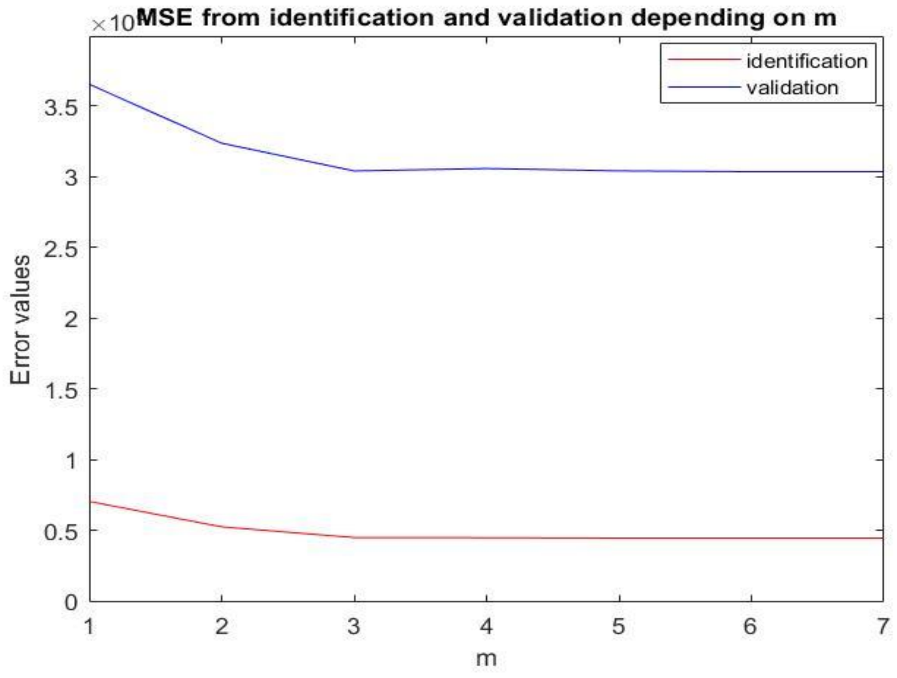
- **Tuning parameters**

The following graphs display the identification and validation MSEs as functions of the Fourier Series variable m:

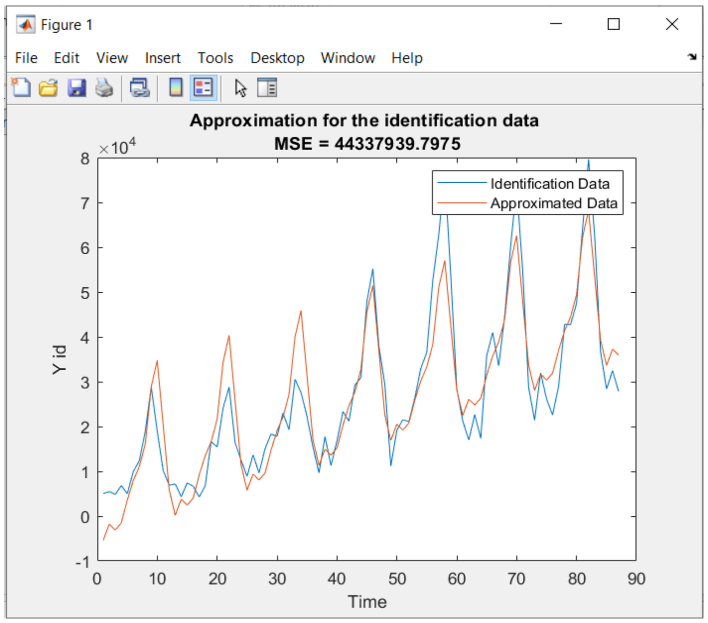


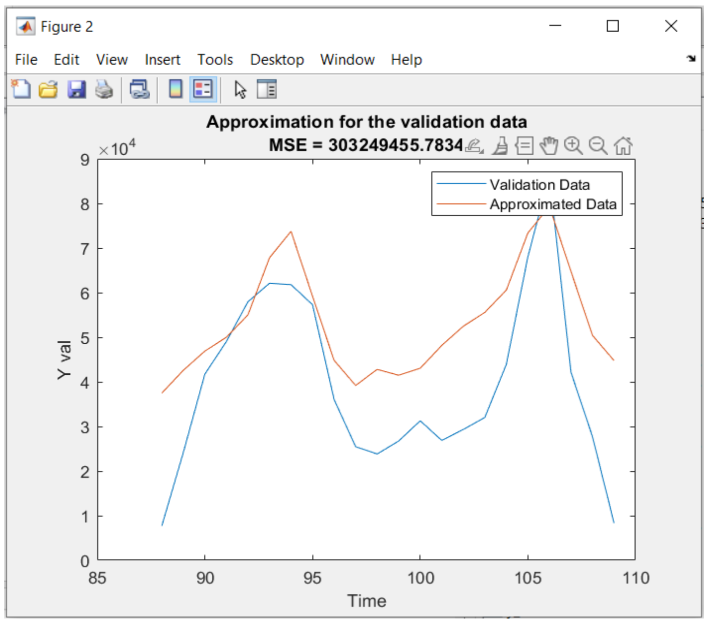


It is observed that the smallest MSEs are for m equal to 6 or 7, where they are equal to $10^7 * 4.4338$ and $10^8 * 3.0325$ respectively.
The next graph displays both MSEs depending on m:



It is also observed that the smallest difference between MSEs is also for m equal to 6 or 7, that difference being equal to $10^8 * 2.5891$. Therefore, the best approximations for our dataset will be looking like in Figure 1 and Figure 2:





- **Conclusions**

  We can conclude that our fitting becomes more precise by increasing the number of Fourier terms “m” and we are obtaining the optimal parameter vector for m equal to 6 or 7. As a result we fulfilled our objective to identify the behavior of the data and successfully model it in order to predict the trend that it follows.

- Below is attached an appendix consisting of 2 pages containing the algorithm of the project, written in MATLAB.

```
clear variables
clc
close all

% Load project data
project_data = load ('product_1.mat');
time = project_data.time;
y = project_data.y;
% 80% of project dataset is identification, 20% is validation.
y_id = y(1:87);
y_val = y(87:109);
time_id = time(1:87);
time_val = time(87:109);
P = 12; % 12 by default (see project document)

%  Initializing the vectors which will contain the MSEs with 0
error_id = zeros(1,7);
error_val = zeros(1,7);

for x = 1:1:7

m = x;
phi_transposed_matrix_id = zeros(length(time_id), 2 + 2 * m);

%   Building the phi transposed matrix for the identification data
for i = 1 : length(time_id)
    phi_transposed_matrix_id(i, 1) = 1;
    phi_transposed_matrix_id(i, 2) = time_id(i);
    for j = 3 : 2 : (2 * m + 1)
        phi_transposed_matrix_id(i, j) = cos(pi * (j-1) * time_id(i) / P);
        phi_transposed_matrix_id(i, j + 1) = sin(pi * (j-1) * time_id(i) / P);
    end
end
%Calculating the parameter vector
theta = phi_transposed_matrix_id \ y_id;

%   Calculating the approximated output
y_approximated_id = phi_transposed_matrix_id * theta;

%   Calculating the Mean Square Error for the identification data
err_squared_id = (y_id - y_approximated_id) .^ 2;
MSE_id = (1 / length(time_id)) * sum(err_squared_id);

error_id(x) = MSE_id;

%   Validating the data

phi_transposed_matrix_val = zeros(length(time_val), 2 + 2 * m);

%   Building the phi transposed matrix for the validation data
for i = 1 : length(time_val)
    phi_transposed_matrix_val(i, 1) = 1;
    phi_transposed_matrix_val(i, 2) = time_val(i);
    for j = 3 : 2 : (2 * m + 1)
        phi_transposed_matrix_val(i, j) = cos(pi * (j-1) * time_val(i) / P);
        phi_transposed_matrix_val(i, j + 1) = sin(pi * (j-1) * time_val(i) / P);
    end
end

%   Calculating the aporiximated output
y_approximated_val = phi_transposed_matrix_val * theta;

%   Calculating the Mean Square Error for the validation data
err_squared_val = (y_val - y_approximated_val) .^ 2;
MSE_val = (1 / length(time_val)) * sum(err_squared_val);

error_val(x) = MSE_val;
end

%   Plotting the approximated output for the identification data
figure,
plot(time_id, y_id, time_id, y_approximated_id);
title("Approximation for the identification data " + newline + "MSE = " + MSE_id),
xlabel("Time");
ylabel("Y id");
legend("Identification Data", "Approximated Data");

%   Plotting the approximated output for the valiation data
figure,
plot(time_val, y_val, time_val, y_approximated_val)
title("Approximation for the validation data"  + newline + "MSE = " + MSE_val),
xlabel("Time");
ylabel("Y val");
legend("Validation Data", "Approximated Data");

%   Plotting the MSEs as functions of m
figure,
plot(error_id)
title('MSE for identification values depending on m')
xlabel('m')
ylabel('Error values')

figure,
plot(error_val)
title('MSE for validation values depending on m')
xlabel('m')
ylabel('Error values')

figure
plot(time_id, y_approximated_id, time_val, y_approximated_val, time, y)
legend("Id Aproximated Data", "Val Approximated Data", "Actual Data");

disp('Minimum MSE_id');
min(error_id)
disp('Minimum MSE_val');
min(error_val)
```